2025-10-24

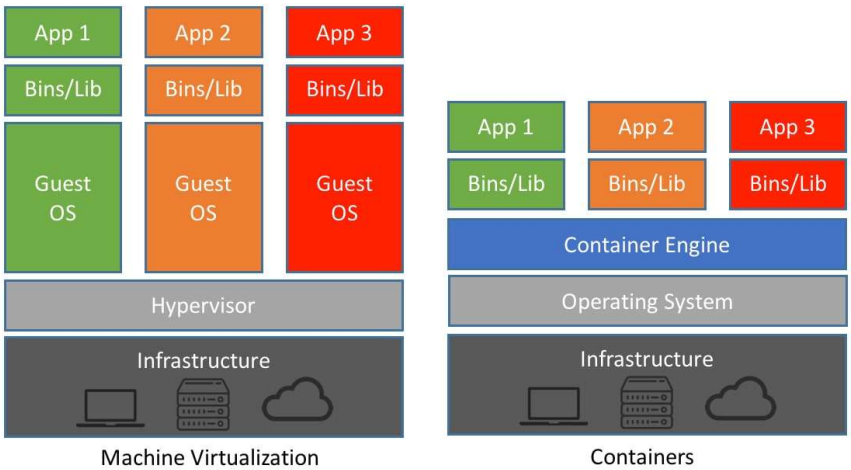
Wednesday, November 26, 2025 11:48 PM

Tema:
Creación y despliegue de una aplicación NestJS utilizando contenedores Docker.

Objetivos:

- Analizar las diferencias fundamentales entre la virtualización tradicional y la tecnología de contenedores para comprender la eficiencia en el uso de recursos.
- Configurar un entorno de desarrollo inicial instalando el framework NestJS y generando una nueva aplicación.
- Construir una imagen de Docker personalizada para una aplicación NestJS y ejecutarla dentro de un contenedor.
- Administrar el ciclo de vida de los contenedores mediante Docker Desktop, gestionando puertos y entornos simulados (desarrollo, pruebas y producción).

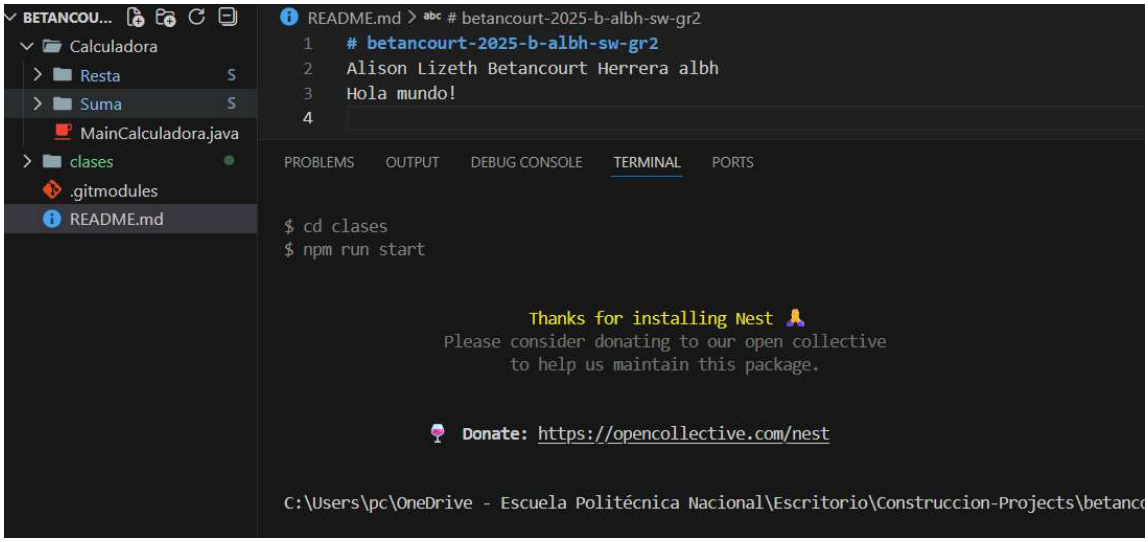
Desarrollo:
Fundamentos: Virtualización vs. Contenedores
Se analizó la arquitectura de los sistemas operativos. En la virtualización tradicional, un **Hipervisor** simula hardware físico, lo que requiere instalar un Sistema Operativo completo (Guest OS) para cada máquina virtual, consumiendo muchos recursos (RAM, CPU, Disco).
En contraste, se estudió cómo **Docker** utiliza un "Container Engine". Esto permite que los contenedores compartan el mismo núcleo (Kernel) del sistema operativo anfitrión (como Linux), eliminando la necesidad de hipervisores pesados y sistemas operativos invitados redundantes. Esto facilita tener el mismo ambiente de desarrollo en diferentes áreas del equipo.



1. Se debe poner la terminal en cmd

2. Para implementar el proyecto con la CLI de Nest, se ejecutó los siguientes comandos, esto creará un nuevo directorio de proyecto y lo llenará con los archivos principales iniciales de Nest y los módulos de soporte, creando así una estructura base convencional para su proyecto.

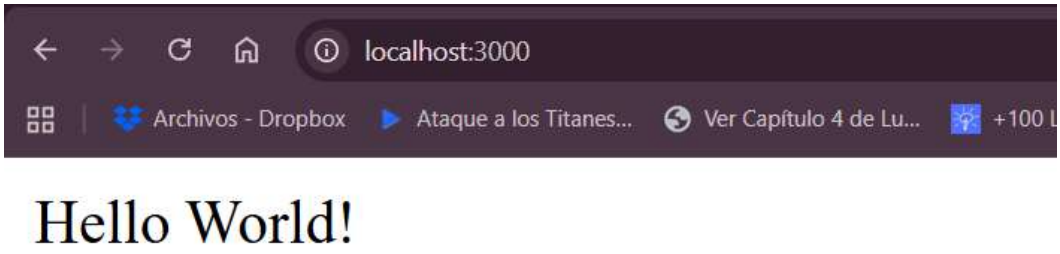
   npm i -g @nestjs/cli
   nest new clases



3. Hay que ingresar a la carpeta y correr el aplicativo con los siguientes comandos:

   cd clases
   npm run start

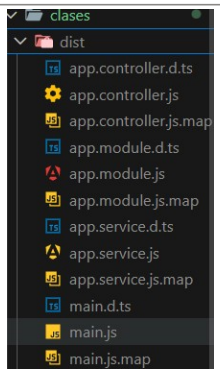
Ahí se ve como al abrir en localhost:3000 el aplicativo está iniciado.



4. Crear una imagen de docker para el aplicativo.

Paso 1: Prepara tu proyecto NestJS
Asegúrate de que tu proyecto esté listo para producción:

- Ejecuta npm run build para compilar tu código TypeScript a JavaScript.
- Verifica que el directorio dist/ se haya creado.
- Tu archivo principal suele estar en dist/main.js.



Paso 2: Crea el archivo Dockerfile
Este archivo le dice a Docker cómo construir la imagen. Aquí tienes un ejemplo básico y explicado:

```
# 1. Usa una imagen base con Node.js
FROM node:22-alpine
# 2. Establece el directorio de trabajo dentro del contenedor
WORKDIR /app
# 3. Copia los archivos necesarios
COPY package*.json ./
# 4. Instala las dependencias
RUN npm install --production
# 5. Copia el resto del código
COPY . .
```

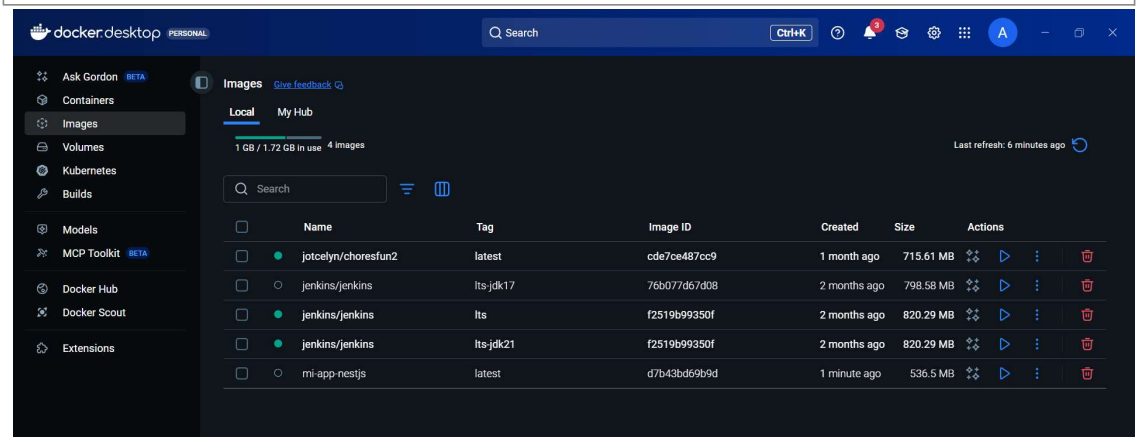
```
# 6. Compila el proyecto NestJS
RUN npm run build
# 7. Expone el puerto en el que corre tu app (por defecto 3000)
EXPOSE 3000
# 8. Comando para iniciar la app
CMD ["node", "dist/main"]
```

Paso 3: Construye la imagen Docker

Abre tu terminal en la raíz del proyecto y ejecuta:

docker build -t mi-app-nestjs .

- -t mi-app-nestjs: le da un nombre a tu imagen.
- . : indica que el Dockerfile está en el directorio actual.



Paso 4: Ejecuta tu contenedor

Una vez creada la imagen, puedes correrla así:

docker run -p 3000:3000 mi-app-nestjs

- -p 3000:3000: expone el puerto 3000 del contenedor en tu máquina local.

Paso 5: Cambiar nombre del contenedor

Para asignarle un nombre personalizado a tu contenedor al momento de ejecutarlo con Docker, puedes usar la opción --name. Aquí tienes el formato:

docker run --name mi-contenedor -p 3001:3000 mi-app-nestjs

Explicación:

- --name mi-contenedor: le da el nombre mi-contenedor al contenedor.
- -p 3001:3000: mapea el puerto 3000 del contenedor al puerto 3001 de tu máquina.
- mi-app-nestjs: es la imagen que estás usando.

Esto te permite referenciar el contenedor fácilmente más adelante, por ejemplo:

docker stop mi-contenedor

docker start mi-contenedor

```
docker logs mi-contenedor
```

Tomar en cuenta que se deben borrar los contenedores que estén ocupando el puerto que se necesita si no son relevantes.

Conclusiones:

- Docker optimiza significativamente el rendimiento en comparación con las máquinas virtuales tradicionales, ya que sus contenedores son ligeros y no requieren cargar un sistema operativo completo para cada aplicación, lo que resulta en un menor consumo de recursos.
- El uso de un Dockerfile permite empaquetar una aplicación NestJS de manera sencilla y estandarizada, garantizando que el software funcione de manera idéntica tanto en la máquina del desarrollador como en el servidor de producción.
- Herramientas como Docker Desktop simplifican la orquestación de contenedores, permitiendo a los desarrolladores gestionar mapeos de puertos y múltiples entornos (Dev, Test, Prod) sin conflictos de configuración ni dependencias externas complejas.

Bibliografia:

Prompts:

Tengo un aplicativo en nodejs que utilliza nestjs como framework. Quisiera saber como podria yo utilizar mi codigo para crear una nueva imagen de docker de mi aplicativo. Me gustaria que me ayudes a entender todo este proceso, porque no lo tengo claro, se lo mas educativo que puedas por favor. Arregla si tengo la versión 22.17.1 para usar el FROM en el dockerfile. Dame bibliografía en formato IEEE para poder entender mejor el tema sobre Creación y Despliegue de Aplicaciones con NestJS y Docker.

[1] M. A. M. Almiron, "NestJS - A Progressive Node.js Framework," *NestJS Documentation*, 2023. [En línea]. Disponible: https://nestjs.com. [Accedido: 17-oct-2025].
[2] Docker, "Docker Overview," *Docker Documentation*, 2023. [En línea]. Disponible: https://docs.docker.com/get-started/. [Accedido: 17-oct-2025].
[3] S. Jones, "Introduction to Docker and Containerization," *Container Journal*, 2023. [En línea]. Disponible: https://containerjournal.com/. [Accedido: 17-oct-2025].
[4] J. Smith, "Understanding Virtualization and Containerization Technologies," *TechRadar*, 2022. [En línea]. Disponible: https://www.techradar.com. [Accedido: 17-oct-2025].
[5] A. D. Johnson, "Building Node.js Applications with Docker," *Docker Blog*, 2021. [En línea]. Disponible: https://www.docker.com/blog. [Accedido: 17-oct-2025].
[6] Docker, "Dockerize an application," *Docker Docs*, 2024. [En línea].

Disponible: https://docs.docker.com/get-started/02_our_app/. [Accedido: 17-oct-2025].